263820
MSWord97/MSEqn3.0
MLW/man
10/22/2001

PATENT APPLICATION
Docket No.: 1086.1007-005

Date: 10/24/01
EXPRESS MAIL LABEL NO. EL906795794US

Inventors: Hong Zhao, Guillermo Sentoni and John Guiver

Attorney's Docket No.: 1086.1007-005

NON-LINEAR DYNAMIC PREDICTIVE DEVICE

RELATED APPLICATIONS

This application is a continuation-in-part of Application No. 09/160,128, filed September 24, 1998, which claims the benefit of U.S. Provisional Application No. 60/060,638 filed October 1, 1997, the contents of which are incorporated herein by reference in their entirety.

I. FIELD OF THE INVENTION

The present invention pertains to a predictive device that models the dynamic input/output relationships of a physical process, particularly in the process industries such as hydrocarbons, polymers, pulp and paper, and utilities. The predictive device is primarily for multivariable process control, but is also applicable to dynamic process monitoring, or to provide a continuous stream of inferred measurements in place of costly or infrequent laboratory or analyzer measurements.

II BACKGROUND OF THE INVENTION

Most existing industrial products designed for multivariable model predictive control (MPC) employ linear step-response models or finite impulse response (FIR) models. These approaches result in over-parameterization of the models (Qin and

Badgwell, 1996). For example, the dynamics of a first order single input/single output SISO process which can be represented with only three parameters (gain, time constant and dead-time) in a parametric form typically require from 30 to 120 coefficients to describe in a step-response or FIR model. This over-parameterization problem is exacerbated for non-linear models since standard non-parametric approaches, such as Volterra series, lead to an exponential growth in the number of parameters to be identified. An alternative way to overcome these problems for non-linear systems is the use of parametric models such as input-output Nonlinear Auto-Regressive with eXogenous inputs (NARX). Though NARX models are found in many case-studies, a problem with NARX models using feed forward neural networks is that they offer only short-term predictions (Su, et al, 1992). MPC controllers require dynamic models capable of providing long-term predictions. Recurrent neural networks with internal or external feedback connections provide a better solution to the long-term prediction problem, but training such models is very difficult.

The approach described in (Graettinger, et al, 1994) and (Zhao, et al, 1997) provides a partial solution to this dilemma. The process model is identified based on a set of decoupled first order dynamic filters. The use of a group of first order dynamic filters in the input layer of the model enhances noise immunity by eliminating the output interaction found in NARX models. This structure circumvents the difficulty of training a recurrent neural network, while achieving good long-term predictions. However, using this structure to identify process responses that are second order or higher can result in over sensitive coefficients and in undesirable interactions between the first order filters. In addition, this approach usually results in an oversized model structure in order to achieve sufficient accuracy, and the model is not capable of modeling complex dynamics such as oscillatory effects. In the single input variable case, this first order structure is a special case of a more general nonlinear modeling approach described (Sentoni et al., 1996) that is proven to be able to approximate any discrete, causal, time invariant, nonlinear SISO process with fading memory. In this approach a Laguerre expansion creates a cascaded configuration of a low pass and several identical band pass first order filters. One of the problems of this approach is that may it require an excessively large degree of expansion to obtain sufficient accuracy. Also, it has not been known until now how to extend this methodology in a practical way to a multi-input system.

This invention addresses many essential issues for practical non-linear multivariable MPC. It provides the capability to accurately identify non-linear dynamic processes with a structure that

- has close to minimum parameterization
- can be practically identified with sufficient accuracy
- makes good physical sense and allows incorporation of process knowledge
- can be proven to identify a large class of practical processes
- can provide the necessary information for process control

III SUMMARY OF THE INVENTION

The present invention is a dynamic predictive device that predicts or estimates values of process variables that are dynamically dependent on other measured process variables. This invention is especially suited to application in a model predictive control (MPC) system. The predictive device receives input data under the control of an external device controller. The predictive device operates in either *configuration* mode or one of three runtime modes - *prediction* mode, *horizon* mode, or *reverse horizon* mode.

The primary runtime mode is the *prediction* mode. In this mode, the input data are such as might be received from a distributed control system (DCS) as found in a manufacturing process. The device controller ensures that a contiguous stream of data from the DCS is provided to the predictive device at a synchronous discrete base sample time. The device controller operates the predictive device once per base sample time and receives the prediction from the output of the predictive device.

After the *prediction* mode output is available, the device controller can switch to *horizon* mode in the interval before the next base sample time. The predictive device can be operated many times during this interval and thus the device controller can conduct a series of experimental scenarios in which a sequence of input data can be specified by the device controller. The sequence of input data can be thought of as a data path the inputs will follow over a forward horizon. The sequence of predictions at the output of the controller is a predicted output path over a prediction horizon and is passed to the device controller for analysis, optimization, or control. The device controller informs the predictive device at the start of an experimental path and synchronizes the presentation of the path with the operation of the device. Internally, *horizon* mode operates exactly the same way as *prediction* mode, except that the dynamic states are maintained separately so that the predictive device can resume normal *prediction* mode operation at the next base sample time. In addition, the outputs of the filter units are buffered over the course of the path and are used during *reverse horizon* operation of the device.

The purpose of *reverse horizon* mode is to obtain the sensitivities of the predictive device to changes in an input path. *Reverse horizon* mode can only be set after *horizon* mode operation has occurred. The device controller first informs the predictive device the index of the point in the output path for which sensitivities are required. The device controller then synchronizes the reverse operation of the predictive device with the output of sensitivity data at the input paths of the device.

In forward operation, each input is scaled and shaped by a preprocessing unit before being passed to a corresponding delay unit which time-aligns data to resolve dead time effects such as pipeline transport delay. Modeling dead-times is an important issue for an MPC system. In practical MPC, prediction horizons are usually set large enough so that both dynamics and dead-time effects are taken into account; otherwise the optimal control path may be based on short term information, and the control behavior may become oscillatory or unstable. In the preferred embodiment, the predictive device is predicting a single measurement, and the dead-time units align data relative to the time of that measurement. If predictions at several measurement points are required, then several predictive devices are used in parallel. During *configuration* mode, the dead times are automatically estimated using training data collected from the plant. In the preferred embodiment the training method consists of constructing individual auto-regressive models between each input and the output at a variety of dead-times, and choosing the dead time corresponding to the best such model. As with other components of the invention, manual override of the automatic settings is possible and should be used if there is additional process knowledge that allows a more appropriate setting.

Each dead time unit feeds a dynamic filter unit. The dynamic filter units are used to represent the dynamic information in the process. Internally the dynamic filter units recursively maintain a vector of states. The states derive their values from states at the previous time step and from the current input value. This general filter type can be represented by what is known to those skilled in the art as a discrete state space equation. The preferred embodiment imposes a much-simplified structure on the filter unit that allows for fast computation for MPC and also allows intelligent override of the automatic settings. This simplified structure is composed of first and second order loosely coupled subfilters, only one of which receives direct input from the corresponding delay unit. The practical identification of this filter structure is an essential part of this invention.

The outputs of the dynamic filter units are passed to a non-linear analyzer that embodies a static mapping of the filter states to an output value. The exact nature of the non-linear analyzer is not fundamental to this invention. It can embody a non-linear mapping such as a Non-linear Partial Least Squares model or a Neural Network, or a hybrid combination of linear model and non-linear model. The preferred embodiment makes use of a hybrid model. The reason for this is that a non-parametric non-linear model identified from dynamic data (such as a neural net) cannot, by its nature, be fully analyzed and validated prior to use. The non-linearity of the model means that different dynamic responses will be seen at different operating points. If the process being modeled is truly non-linear, these dynamic responses will be an improvement over linear dynamic models in operating regions corresponding to the training data, but may be erroneous in previously unseen operating regions. When the non-linear model is used within the context of MPC, erroneous responses, especially those indicating persistent and invalid gain reversals can create instabilities in the MPC controller. With a hybrid approach, a non-linear model is used to model the errors between the linear dynamic model and the true process. The hybrid dynamic model is a parallel combination of the linear dynamic model with the error correction model. The dynamic response of the linear model can be analyzed completely prior to use, since the gains are fixed and independent of the operating point. The process engineer can examine and approve these gains prior to closing the loop on the process and is assured of responses consistent with the true process. However, the linear dynamic response will be sub-optimal for truly non-linear processes. In online operation of the hybrid model within an MPC framework, the responses of the linear model and the hybrid model can be monitored independently and compared. In operating regions where the non-linear model shows persistently poor response, control can be switched, either automatically or by the operator, back to the safety of the linear model.

The output of the non-linear analyzer is passed through a postprocessing unit that converts the internal units to engineering units.

The importance of this invention is that its structure is shown to be able to approximate a large class of non-linear processes (any discrete, causal, time invariant, nonlinear multi-input/single output (MISO) process with fading memory), but is still simple enough to allow incorporation of process knowledge, is computationally fast enough for practical non-linear MPC, and can be configured with sufficient accuracy in a practical manner.

IV BRIEF DESCRIPTION OF THE DRAWINGS

The textual description of the present invention makes detailed reference to the following drawings:

FIG. 1 is an overall block diagram of the invention showing both the runtime and training components.

FIG. 2 shows the runtime structure of an individual preprocessing unit.

FIG. 3 shows the runtime structure of an individual delay unit.

FIG. 4 shows the forward flow internal decomposition of an individual filter unit into cascaded subfilter units.

FIG. 5 shows the preferred forward flow structure of a primary first order subfilter unit.

FIG. 6 shows the preferred forward flow structure of a secondary first order subfilter unit and the preferred coupling with the previous subfilter in the cascade.

FIG. 7 shows the preferred forward flow structure of a primary second order subfilter unit.

FIG. 8 shows the preferred forward flow structure of a secondary second order subfilter unit and the preferred coupling with the previous subfilter in the cascade.

FIG. 9 shows a typical feedforward configuration of the non-linear analyzer.

FIG.10 shows the reverse flow configuration of the non-linear analyzer depicted in FIG. 9.

FIG.11 shows the reverse flow internal decomposition of an individual filter unit into cascaded subfilter units.

FIG. 12 shows a method of training an individual delay unit.

FIG. 13 shows the first order decoupled structure used at the start of each iteration of the preferred dynamic filter unit identification method.

FIG. 14 shows that reverse flow of data through a matrix structure can be described mathematically by forward flow of data through the transpose matrix structure.

V DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG.1 is an overall block diagram of the invention and its context. An external device controller (**50**) synchronizes the flow of data to and from the predictive device via the data paths (**18**), (**14**), and (**64**). The device controller also controls the mode of operation and the path stepping of the predictive device via the control path (**54**). The external device controller may also communicate with a DCS (**10**) or other data/control system both for requesting data and for requesting control changes to the modeled process; however the exact external context and configuration of the device controller is beyond the scope of this application.

V.1 FORWARD RUNTIME OPERATION OF THE PREDICTION DEVICE

The figures and equations in this detailed description refer to an index k that represents a data point in a sequence of data points. This index has different meanings depending on whether the forward operational mode of the device is *prediction* mode or *horizon* mode.

In *prediction* mode data is provided at a regular sampling interval Δt to the input nodes **(18)** of the device. Data is passed in a forward direction through the device. For simplicity of notation, the sample point $T_0+k\Delta t$ is denoted by the index k.

In *horizon* mode, a sequence of data representing a forward data path is provided to the inputs. This data path may represent a proposed path for manipulated variables for process control purposes, or may represent a holding of the inputs to constant values in order to determine the steady state output of the device. The starting point of this path is taken to be the most recent input sample provided in *prediction* mode. Index *0* represents this starting point and index k represents the k^{th} data point in this path.

V.1.1 FORWARD RUNTIME OPERATION OF A PREPROCESSING UNIT

Each input feeds a preprocessing unit **(20)** which is used to convert the engineering units of each data value to a common normalized unit whose lower and upper limits are, by preference, -1 and 1 respectively, or 0 and 1 respectively.

The preprocessing unit can also shape the data by passing it through a non-linear transformation. However, the preferred embodiment uses a simple scale and offset as shown in *FIG. 2* and equation (1):

$$u(k) = su_E(k) + o \tag{1}$$

where $u_E(k)$ is the value of an input in engineering units, and $u(k)$ is the preprocessed value in normalized units. The scale and offset values as stored in the configuration file (**30** - *FIG. 1*) are, in general, different for each input variable, and are determined in the *configuration* mode.

V.1.2 FORWARD RUNTIME OPERATION OF A DELAY UNIT

Data flows from each preprocessing unit to a corresponding delay unit (**22**). The forward run-time operation of the delay unit (**22**) is shown in *FIG 3* and equation (2). The output $u^d(k)$(**304**) of an individual delay unit (**300**) is equal to the input $u(k)$ (**302**) delayed by d sample times. The value of d may be different for each delay unit (**22**) and is retrieved from the configuration file (**30** - *FIG. 1*). This may be implemented as a shift register with a tap at the d^{th} unit.

$$u^d(k) = u(k-d) \qquad (2)$$

This equation can also be written in terms of the unit delay operator q^{-1}:

$$u^d(k) = q^{-d}u(k)$$

V.1.3 FORWARD RUNTIME OPERATION OF THE FILTER UNITS

Referring again to *FIG. 1*, each delayed input value is passed to an individual filter unit (**24**). The general internal feedforward structure of a filter unit (**24**) is shown in *FIG. 4*. The general feedforward structure is composed of S cascaded subfilters (**402**, **404**, ..., **406**). The first subfilter in the cascade (**400**) is referred to as the **primary subfilter**. Non-primary subfilters are referred to as **secondary subfilters**. All the subfilters are alike except that the primary subfilter receives no input from another subfilter, and the final subfilter sends no output to another subfilter. Now the general form of the primary subfilter will be described in detail.

The primary subfilter maintains a vector (**412**) of states $\boldsymbol{x}_1(k)$ at each time k. An internal single time step delay unit (**414**) feeds the vector state to a coupling unit (**420**) and to a matrix unit (**416**). The matrix unit converts the delayed state vector (**418**) and feeds it to a vector addition unit (**408**). The input to the filter unit *$u^d(k)$* is expanded and linearly scaled by the input coupling unit (**410**) to a vector of values of the same dimension as the state vector. The vector addition unit then combines its two input streams to produce the vector of states for the current time. The operation just described for the primary subfilter is conveniently described in mathematical matrix and column vector notation as:

$$\mathbf{x}_1(k) = \mathbf{A}_1\mathbf{x}_1(k-1) + \mathbf{b}_1 u^d(k) \qquad (3)$$

Such an equation is known, to those skilled in the art, as a linear state space equation with a single input. If no structure is imposed on $\mathbf{A}_1$ or $\mathbf{b}_1$, then further subfilters are unnecessary since the cascaded subfilter structure can subsumed into a single complicated primary subfilter. However, the preferred subfilter structures as described below, or similar to those described below, are essential for a practical embodiment and application of the invention.

The subfilter coupling unit (**420**) determines how state values at time *k-1* affect the state units in the next subfilter at time *k*. In mathematical terms, the subfilter coupling unit uses the coupling matrix Γ_2 to perform a linear transformation of state vector $\boldsymbol{x}_1(k-1)$ which is passed to the vector addition unit of the next subfilter. The operation of a secondary subfilter is conveniently described in mathematical matrix and vector notations as:

$$\mathbf{x}_s(k) = \mathbf{A}_s\mathbf{x}_s(k-1) + \mathbf{\Gamma}_s\mathbf{x}_{s-1}(k-1) + \mathbf{b}_s u^d(k) \qquad (4)$$

In the preferred embodiment, the subfilters are all of first or second order. A first order subfilter maintains just one state. The preferred embodiment for a first order primary subfilter (**500**) is shown in *FIG. 5*. The vectorizing unit (**502**) and the matrix unit (**504**) collapse to become scaling operations so that the state vector (**506**) is represented by:

$$x_1(k) = \lambda_1 x_1(k-1) + (1-\lambda_1)u^d(k) \qquad (5)$$

The preferred embodiment for a first order secondary subfilter **(600)** is shown in *FIG. 6.* The secondary subfilter receives no direct input, but instead receives cascaded input from the previous subfilter. The preferred coupling is a loose coupling scheme **(602)** in which only the last state component of the previous subfilter contributes. Note that the previous subfilter is not required to be a first order subfilter. The state vector **(606)** is represented by:

$$x_s(k)=\lambda_s x_s(k-1)+(1-\lambda_s)x_{s-1,last}(k-1) \qquad (6)$$

where the matrix unit λ_s **(604)** is a scalar.

Second order subfilters maintain two states. The preferred embodiment for a second order primary subfilter **(700)** is shown in *FIG. 7.* In this figure, the state vector $\boldsymbol{x_1(k)}$ is shown in terms of its two components $x_{11}(k)$ **(708)** and $x_{12}(k)$ **(710)**. The vectorizing unit **(702)** creates two inputs to the vector addition unit **(714)**, the second of which is fixed at zero. The delayed states **(704)** and **(706)** are fed to the matrix unit **(712)** whose outputs are also fed to the vector addition unit **(712)** which adds the matrix transformed states to the vectorized inputs producing the current state. Note that due to the (1,0) structure of the second matrix row, and the zero second component of the vectorizing unit component, the current second state component **(710)** is just equal to the delayed first component **(704)**:

$$\begin{aligned} x_{11}(k)&=a_{11}x_{11}(k-1)+a_{12}x_{12}(k-1)+(1-a_{11}-a_{12})u^d(k) \\ x_{12}(k)&=x_{11}(k-1) \end{aligned} \qquad (7)$$

The preferred embodiment for a second order secondary subfilter **(800)** is shown in *FIG. 8.* In this figure, the state vector $\boldsymbol{x_s(k)}$ is shown in terms of its two components $x_{s1}(k)$ **(808)** and $x_{s2}(k)$ **(810)**. The preferred coupling with the previous subfilter unit is a loose coupling scheme **(802)** in which only the last state component of the previous subfilter contributes to the first state component of the current subfilter. Note that the previous subfilter is not required to be a first order subfilter or second order subfilter. The output of the coupling unit is fed to the addition unit **(814)**. The delayed states **(804)** and

(806) are fed to the state matrix unit **(812)** whose outputs are also fed to the vector addition unit **(812)** which adds the state matrix transformed states to the output of the coupling unit, producing the current state. Note that due to the (1,0) structure of the second state matrix row, and the zero second row of the coupling matrix, the current second state component **(810)** is just equal to the delayed first component **(804)**:

$$\begin{aligned} x_{s1}(k) &= a_{s1}x_{s1}(k-1) + a_{s2}x_{s2}(k-1) + (1 - a_{s1} - a_{s2})x_{s-1,last}(k-1) \\ x_{s2}(k) &= x_{s1}(k-1) \end{aligned} \qquad (8)$$

If the device is operating in *horizon* mode current states along the path are maintained in a separate storage area so as not to corrupt the *prediction* mode states. In *horizon* mode, *k* indexes the input path and the states are initialized at the start of the path (*k =0*) to the *prediction* mode states. In addition the states at the output of the filter unit are buffered for use in *reverse horizon* mode.

V.1.4 FORWARD RUNTIME OPERATION OF THE NON-LINEAR ANALYZER

Referring again to *FIG. 1*, the outputs **(28)** of the filter units **(24)** provide input to the non-linear analyzer **(26)**. The exact structure and configuration of the non-linear analyzer **(26)** is not central to this application. It is the interaction of the non-linear analyzer **(26)** with the filter units **(24)**, and the operation and configuration of the filter units **(24)** that forms the core of this invention. The preferred embodiment, for reasons discussed in the summary of the invention is a hybrid parallel combination of linear and non-linear. However, for clarity of explanation, a standard neural network structure is described which is well known to those skilled in the art. This structure is shown in *FIG 9*. The equations for this structure are:

$$\begin{aligned} \xi_h(k) &= w_{h0} + \sum_{i=1}^{N} w_{hi}x_i(k) \\ \eta_h(k) &= \tanh(\xi_h(k)) \\ y(k) &= \sum_{h=1}^{H} c_h\eta_h(k) \end{aligned} \qquad (9)$$

V.1.5 FORWARD RUNTIME OPERATION OF THE POSTPROCESSING UNIT

The postprocessing unit (**32**) in *FIG. 1* is used to scale the output from the normalized units to engineering units. The postprocessing unit can also shape the data by passing it through a non-linear transformation. However, the preferred embodiment uses a simple scale and offset. For consistency with the preprocessing units, the scale and offset represent the mapping from engineering units to normalized units.

$$y_E(k) = \frac{1}{s}y(k) - \frac{o}{s} \tag{10}$$

The scale and offset values as stored in the configuration file (**30** - *FIG. 1*) and are determined in the *configuration* mode.

V.2 REVERSE RUNTIME OPERATION OF THE PREDICTION DEVICE

The *reverse horizon* mode of operation is only allowed immediately following *horizon* mode operation. *Horizon* mode operation buffers the states (**28**) output by the filter units (**24**) over the course of the forward path. The purpose of *reverse horizon* mode is to obtain the sensitivity of any point *y(k)* of the prediction path (output by the device in *horizon* mode) with respect to any point in the input path ***u(l)***.

In order to use the invention for process control applications, the mathematical derivatives of the prediction with respect to the inputs are required. The mathematical derivatives measure how sensitive a state is in response to a small change in an input. The dynamic nature of the predictive device means that a change in input at time *k* will start to have an effect on the output as soon as the minimum dead-time has passed and will continue to have an effect infinitely into the future. In most practical applications systems are identified to have fading memory so that the effect into the future recedes with time. For MPC applications the aim is to plan a sequence of moves for the inputs corresponding to manipulated variables (MVs). The effect of these moves needs to be predicted on the controlled variables (CVs) along a prediction path. A constrained optimization algorithm is then used to find the move sequences that predict an optimal prediction path according to some desired criteria.

In *reverse horizon* mode, the external device controller specifies the output path index k. The device then outputs in sequence the sensitivities **(64)** in reverse order at the input nodes of the device. In the detailed description below, the sensitivity of the output $y_E(k)$ of the device with respect to any variable v is represented by $\Omega_k v$ .It is this sensitivity value, rather than an external data value that is fed back through the device when operating in *reverse horizon* mode.

V.2.1 REVERSE RUNTIME OPERATION OF THE POSTPROCESSING UNIT

The reverse operation of the postprocessing unit **(32)** is to scale data received at its output node using the inverse of the feedforward scaling shown in equation (10):

$$\Omega_k y(k) = s\Omega_k y_E(k) \tag{11}$$

Since the sensitivity of the output with respect to itself is:

$$\Omega_k y_E(k) = 1 \tag{12}$$

the postprocessing unit always receives the value of 1 at its output node in reverse operation.

V.2.2 REVERSE RUNTIME OPERATION OF THE NON-LINEAR ANALYZER

The reverse runtime operation of a neural net model is well known to those skilled in the art and is shown in *FIG. 10.* The output from the reverse operation of the postprocessing unit $\Omega_k y(k)$ is presented at the output node of the non-linear analyzer **(26)**. The information flows in a reverse manner through the non-linear analyzer **(26)** and the resulting sensitivities **(62)** are output at the input nodes of the non-linear analyzer **(26)**:

$$\begin{aligned}
\Omega_k \eta_h(k) &= c_h \Omega y(k) \\
\Omega_k \xi_h(k) &= \Omega_k \eta_h(k) \tanh'(\xi_h(k)) \\
&= \Omega_k \eta_h(k)(1 - \eta_h(k))(1 + \eta_h(k)) \\
\Omega_k x_i(k) &= \sum_{h=1}^{H} w_{hi} \Omega_k \xi_h(k)
\end{aligned} \tag{13}$$

V.2.3 REVERSE RUNTIME OPERATION OF A FILTER UNIT

The effect of a change in the delayed input $u^d(l)$ on a the sequence of states being output from a filter unit (**24**) in *horizon* mode is complex due to the dependencies of a subfilter's states based on the previous subfilter's states and on the subfilter's previous states. An efficient solution can be derived using the chain rule for ordered derivatives (Werbos, 1994) and is achieved by the reverse operation of the filter unit (**24**). In *reverse horizon* mode, the output of each filter unit (**24**) receives the vector of sensitivities $\Omega_s \mathbf{x}_s(k)$ propagated back from the non-linear analyzer (**26**) operating in reverse mode:

$$\Omega_k \mathbf{x}_s(l) = \begin{cases} \Omega_k \mathbf{x}_s(k) & l = k \\ \mathbf{A}_s^T(\Omega_k \mathbf{x}_s(l+1)) + \Gamma_{s+1}^T(\Omega_k \mathbf{x}_{s+1}(l+1)) & l < k, 1 \le s < S \\ \mathbf{A}_S^T(\Omega_k \mathbf{x}_S(l+1)) & l < k, s = S \\ 0 & l > k \end{cases}$$

$$\Omega_k u^d(l) = \sum_{s=1}^{S} \mathbf{b}_s^T \Omega_k \mathbf{x}_s(l) \tag{14}$$

The operation of these equations is shown in *FIG. 11*, which shows the filter structure of *FIG. 4,* but with data flowing in the reverse direction. Given the point k in the output path for which the sensitivities are being calculated, the vector of sensitivities $\Omega_k \mathbf{x}_s(k)$ is presented at the output channels (**1120, 1122...1124**) of the filter unit (**24**) and cycled in reverse through the filter structure. This reverse operation is indexed by $l \le k$. At each iteration l, the resulting sensitivity $\Omega_k u^d(l)$ is output at the input channel (**1110**) of the filter unit (**24**). For $l < k$ the external input at the output channels (**1120, 1122...1124**) is in practice zero vector since $\Omega_k \mathbf{x}_s(l) = 0$. However, the filter unit (**24**) itself is not constrained to operate under this assumption.

In *FIG. 11*, the reverse operation of a delay (**1130**) is represented by q which is the unit delay in the reverse time direction since the index l is decreasing at each iteration.

The reverse operation of a matrix operation (**1132, 1134**) or a vector operation (**1136**) is represented mathematically as the transpose of the forward operation. The physical justification for this is shown in *FIG. 14* which shows the individual channels represented by a 3x2-matrix operation which in forward operation maps two input channels to three output channels, and in reverse operation maps three input channels to two output channels.

V.2.4 REVERSE RUNTIME OPERATION OF A DELAY UNIT

The reverse operation of a delay unit (**22**) corresponds to a delay in the reverse sequencing:

$$\Omega_k u(l) = \Omega_k u^d(l+d) \tag{15}$$

V.2.5 REVERSE RUNTIME OPERATION OF A PREPROCESSING UNIT

The reverse operation of a preprocessing unit (**20**) is to scale data received at its output node using the inverse of the feedforward scaling shown in equation (1):

$$\Omega_k u_E(l) = \frac{1}{s}\Omega_k u(l) \tag{16}$$

V.3 CONFIGURATION MODE

The predictive device is configured, in the preferred embodiment, using training data collected from the process. However, a process engineer can override any automated configuration settings. The training data set should represent one or more data sets which have been collected at the same base-time sample rate that will be used by the external device controller to present data to the predictive device in *prediction* mode. Each set of data should represent a contiguous sequence of representative.

In order to allow operator approval or override of the configuration settings, the training of the predictive device is done in stages, each stage representing a major component of the predictive device.

V.3.1 CONFIGURING THE PREPROCESSING AND POSTPROCESSING UNITS

The scale and offset of a preprocessing or postprocessing unit is determined from the desire to map the minimum E_{min} and maximum E_{max} of the corresponding variable's engineering units to the minimum N_{min} and maximum N_{max} of the normalized units:

$$s = \frac{N_{\max} - N_{\min}}{E_{\max} - E_{\min}} \qquad o = \frac{E_{\max} N_{\min} - E_{\min} N_{\max}}{E_{\max} - E_{\min}} \tag{17}$$

The preferred normalized units have N_{min}=-1, N_{max}=1. The engineering units may be different for each input variable, leading to a different scale and offset for each preprocessing/postprocessing unit.

V.3.2 CONFIGURING A DELAY UNIT

The configuration of a delay unit (**22**) is not a central aspect of this application. *FIG. 12* shows a simple advisory procedure for suggesting delay times. A process engineer can override these advisory settings. In this procedure $d_{\min}$ and $d_{\max}$ are user settable limits for the delay time and the procedure calculates a delay time d such that

$$d_{\min} \le d \le d_{\max}$$

V.3.3 CONFIGURING A FILTER UNIT

A practical means of configuring a filter unit (**24**) is an essential aspect of this invention. The preferred method of configuration is initialized using the simplified filter structure shown in *FIG. 13* in which all subfilters are first order and decoupled. This is the structure used in (Graettinger, et al, 1994). It is important to note that this structure is used for initialization of the configuration procedure and does not represent the final suggested filter configuration.

Step 1

The operator specifies an appropriate dominant time constant T_i associated with each input variable. This can be specified from engineering knowledge or through an automated approach such as Frequency Analysis or a Back Propagation Through Time algorithm. The value of the initial time constant is not critical the proposed configuration method automatically searches the dominant time range for the best values.

Step 2

For each input, initialize the filter structure in *FIG. 13* using a high order system where a number of first order filters are created around the given dominant time constant (dominant frequency, dominant dynamics). For example, a fifth order system can be created using:

$$\begin{aligned} \lambda_{i1} &= e^{-\frac{\Delta t}{0.5T_i}} \\ \lambda_{i2} &= e^{-\frac{\Delta t}{0.75T_i}} \\ \lambda_{i3} &= e^{-\frac{\Delta t}{T_i}} \\ \lambda_{i4} &= e^{-\frac{\Delta t}{1.25T_i}} \\ \lambda_{i5} &= e^{-\frac{\Delta t}{1.5T_i}} \end{aligned} \tag{18}$$

In this simple filter structure, each subfilter (**1302, 1304, 1306**) yields a corresponding single state (**1312, 1314, 1316**) which is decoupled from the other subfilter states. This initial filter structure represents the equation

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \mathbf{B}\mathbf{u}^d(k) \tag{19}$$

which has a simplified diagonal block structure of the form

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \\ \vdots \\ \mathbf{x}_N(k) \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & 0 & 0 & 0 \\ 0 & \mathbf{A}_2 & 0 & \vdots \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \mathbf{A}_N \end{bmatrix} \qquad (20)$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & 0 & 0 & 0 \\ 0 & \mathbf{b}_2 & 0 & \vdots \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \mathbf{b}_N \end{bmatrix}$$

where

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{i5} \end{bmatrix}$$

$$\mathbf{A}_i = \begin{bmatrix} \lambda_{i1} & 0 & 0 & 0 \\ 0 & \lambda_{i2} & 0 & \vdots \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_{i5} \end{bmatrix} \qquad (21)$$

$$\mathbf{b}_i = \begin{bmatrix} 1-\lambda_{i1} \\ 1-\lambda_{i2} \\ \vdots \\ 1-\lambda_{i5} \end{bmatrix}$$

Step 3

Map the contiguous input training data through the delay units (**22**) and filter structure (**24**) to obtain a set of training state vectors $\{\mathbf{X}(k) | k = 1, \cdots, T\}$. Then find a vector **c** that provides the best linear mapping of the states to the corresponding target outputs $\{Y(k) | k = 1, \cdots, T\}$. One way of doing this is to use the Partial Least Squares method that is well known to those skilled in the art. This results in a multi-input, single-output (MISO) state space system $\{\mathbf{A},\mathbf{b},\mathbf{c}^T\}$ in which equations (19), (20), and (21) are supplemented by the equation:

$$y(k)=\mathbf{c}^T\mathbf{x} \tag{22}$$

where

$$\mathbf{c}=\begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\\ \vdots\\ \mathbf{c}_N\end{bmatrix},\qquad \mathbf{c}_i=\begin{bmatrix}c_{i1}\\ c_{i2}\\ \vdots\\ c_{i5}\end{bmatrix} \tag{23}$$

Step 4

Balance each subsystem $\{\mathbf{A}_i,\mathbf{b}_i,\mathbf{c}_i^T\}$of the MISO block diagonal system based on controllability & observability theory. The balancing procedure allows order reduction of a state space system by transforming the states so that the controllability and observability properties of the original system are substantially concentrated in the first part of the state vector.

For each input variable, indexed by i, perform the balancing procedure on the sub-system $\{\mathbf{A}_i,\mathbf{b}_i,\mathbf{c}_i^T\}$. Balancing of a linear state space system is a method of reduction well known to those skilled in the art. Other methods of model reduction, such as Hankel reduction, can be substituted. A summary of the balancing method is now given.

For each sub-system $\{\mathbf{A}_i,\mathbf{b}_i,\mathbf{c}_i^T\}$, compute the controllability and observability Gramians $\mathbf{P}_i>0$, $\mathbf{Q}_i>0$ that satisfy the equations:

$$\begin{aligned}\mathbf{A}_i\mathbf{P}_i\mathbf{A}_i^T-\mathbf{P}_i&=-\mathbf{b}_i\mathbf{b}_i^T\\ \mathbf{A}_i^T\mathbf{Q}_i\mathbf{A}_i-\mathbf{Q}_i&=-\mathbf{c}_i\mathbf{c}_i^T\end{aligned} \tag{24}$$

Find a matrix $\mathbf{R}_i$, using the Cholesky factorization method, such that

$$\mathbf{P}_i=\mathbf{R}_i^T\mathbf{R}_i. \tag{25}$$

Using the singular value decomposition method, diagonalize to obtain the following decomposition:

$$\mathbf{R}_i\mathbf{Q}_i\mathbf{R}_i^T = \mathbf{U}_i\Sigma_i^2\mathbf{U}_i^T \tag{26}$$

Define

$$\mathbf{T}_i^{-1} = \mathbf{R}_i^T\mathbf{U}_i\Sigma_i^{-1/2} \tag{27}$$

then

$$\mathbf{T}_i\mathbf{P}_i\mathbf{T}_i^T = \left(\mathbf{T}_i^T\right)^{-1}\mathbf{Q}_i\mathbf{T}_i^{-1} = \Sigma_i \tag{28}$$

and the balanced subsystem is obtained through a similarity transform on the states as:

$$\hat{\mathbf{A}}_i = \mathbf{T}_i\mathbf{A}_i\mathbf{T}_i^{-1}, \hat{\mathbf{b}}_i = \mathbf{T}_i\mathbf{b}_i, \hat{\mathbf{c}}_i^T = \mathbf{c}_i^T\mathbf{T}_i^{-1} \tag{29}$$

Step 5

Using balanced subsystems find out dominant time constant for each input by reducing each balanced model to a first order model. This is done by considering the dynamics of all but the first state of each input's filter unit (**24**) to have reached steady state. This leads to:

$$T_i = -\frac{\Delta t}{\ln(a_i)} \tag{30}$$

where

$$a_i = \hat{a}_{i11} + \hat{\mathbf{a}}_{i12}^T\left(\mathbf{I} - \hat{\mathbf{A}}_{i22}\right)^{-1}\hat{\mathbf{a}}_{i21} \tag{31}$$

and

$$\hat{\mathbf{A}}_i \equiv \begin{bmatrix} \hat{a}_{i11} & \hat{\mathbf{a}}_{i12}^T \\ \hat{\mathbf{a}}_{i21} & \hat{\mathbf{A}}_{i22} \end{bmatrix} \tag{32}$$

Check the convergence of the dominant time constant estimation:

If

$$\frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(a_i^{current} - a_i^{previous}\right)^2} < \varepsilon \qquad (33)$$

or the number of iterations has exceeded the maximum allowable, go to step 6. Otherwise, return to step 2. The maximum number of iterations and ε are parameters of the training method.

Step 6

Once an accurate estimate of the dominant time constant is available for each input variable, the eigenvalues $\{\lambda_{is}^{P} | s = 1,\cdots,5\}$ of the controllability gramian $\hat{\mathbf{P}}_i$ (equivalently the observability gramian) are calculated; these are always positive and real because the controllability gramian is positive definite. The final order S_i of each filter unit (**24**) is then calculated such that

$$\frac{\sum_{s=1}^{S_i-1}\lambda_{is}^{P}}{\sum_{s=1}^{5}\lambda_{is}^{P}} < \theta \leq \frac{\sum_{s=1}^{S_i}\lambda_{is}^{P}}{\sum_{s=1}^{5}\lambda_{is}^{P}} \qquad (34)$$

where θ is parameter of the training method and is a value less than 1, a good practical value being 0.95. This order represents the total number of states of an individual filter unit (**24**).

After determining the model order, truncate the $\hat{\mathbf{A}}_i$ matrix so that just the first S_i states are used; this truncation is done by selecting the upper left S_ix S_i submatrix of $\hat{\mathbf{A}}_i$. Then calculate the S_i eigenvalues of the truncated $\hat{\mathbf{A}}_i$ matrix $\{\lambda_{is} | s = 1,\cdots,S_i\}$. Now configure each filter unit (**24**) using the preferred first and second order subfilter configurations with the preferred couplings as shown in *FIG 5* through *FIG. 8*. Use a first order filter for each real eigenvalue. Use a second order filter for each pair of complex eigenvalues $\{\lambda,\bar{\lambda}\}$, where, in *FIG. 7* (equation 7) or *FIG. 8* (equation 8):

$$\begin{aligned} a_{11} &= \lambda + \bar{\lambda} \\ a_{12} &= -\lambda\bar{\lambda} \end{aligned} \qquad (35)$$

The preferred ordering of these subfilter units is according to time-constant, with the fastest unit being the primary subfilter.

Another favored approach is to perform model reduction by initializing with Laguerre type filter units as described in section V.4.2, rather than the simple diagonal filter structure of *FIG. 13*. Sufficient quantity of Laguerre type filter units span the full range of dynamics in the process, and thus the iterative process described above is not needed. In fact a non-linear model reduction can be achieved by performing a linear model reduction on the linear system whose states are defined by the Laguerre filters and whose outputs are defined by pre-transformed values at the hidden layer of the neural net:

$$\xi_1(k), \cdots, \xi_H(k)$$

V.3.4 CONFIGURING THE NON-LINEAR ANALYZER

The configuration of the non-linear analyzer (**26**) is not a central aspect of this application. The non-linear analyzer (**26**) is trained to optimally map the outputs of the filter units (**24**) to the corresponding target output. Training of a neural net is described in detail in (Bishop, 95) for example. In one embodiment, the non-linear analyzer is replaced by apparatus for a constrained non-linear approximator disclosed in U.S. Patent Application No. 09/892,586 and herein incorporated by reference. In particular, page 33, lines 26-29, as filed in Aplication No. 09/392,586 describes the additional calculations needed when the inputs to the non-linear approximator are filtered states.

V.4 UNIVERSALITY OF THE PREDICTION DEVICE

The predictive device is shown, in this section, to be able to approximate any *time invariant*, *causal*, *fading memory* system (defined below). In order to prove this, some precise notation and definitions will be needed.

V.4.1 NOTATION AND DEFINITIONS FOR UNIVERSALITY PROOF

Let $\mathbf{Z}$ denote the integers, $\mathbf{Z}_+$ the non-negative integers and $\mathbf{Z}_-$ the non-positive integers respectively. A variable $\mathbf{u}$ represents a vector or a sequence in accordance with the context, while $\mathbf{u}(k)$ represents a value of the sequence at the particular time k.

For any positive integer $p > 0$, $\mathbf{R}^N$ denotes the normed linear space of real N-vectors (viewed as column vectors) with norm $|\mathbf{u}| = \max_{1\le i\le N}|u_n|$. Matrices are denoted in uppercase bold. Functions are denoted in italic lowercase if they are scalars and in bold if they are vector valued.

Let $l_N^\infty(\mathbf{Z})$ (respectively $l_N^\infty(\mathbf{Z}_+)$ and $l_N^\infty(\mathbf{Z}_-)$), be the space of bounded $\mathbf{R}^N$-valued sequences defined on $\mathbf{Z}$ (respectively $\mathbf{Z}_+$ and $\mathbf{Z}_-$) with the norm:

$$\|\mathbf{u}\|_\infty = \sup_{k\in\mathbf{Z}}|\mathbf{u}(k)|$$

For every decreasing sequence $w:\mathbf{Z}_+ \to (0,1]$, $\lim_{k\to\infty} w(k) = 0$ define the following weighted norm:

$$\|\mathbf{u}\|_w = \sup_{k\in\mathbf{Z}_-}|\mathbf{u}(k)|w(-k)$$

A function $F : l_N^\infty(\mathbf{Z}_-) \to \mathbf{R}$ is called a *functional* on $l_N^\infty(\mathbf{Z}_-)$, and a function $\Im : l_N^\infty(\mathbf{Z}_-) \to l^\infty(\mathbf{Z})$ is called an *operator*. As a notational simplification the parentheses around the arguments of functionals and operators are usually dropped; for example, $F\mathbf{u}$ rather than $F[\mathbf{u}]$ and $\Im\mathbf{u}(k)$ rather than $\Im[\mathbf{u}](k)$.

Two specific operators are important. The delay operator defined by

$$Q^d\mathbf{u}(k) \equiv \mathbf{u}(k-d)$$

and the truncation operator defined by

$$P\mathbf{u}(k) \equiv \begin{cases} \mathbf{u}(k) & k \le 0 \\ \mathbf{0} & k > 0 \end{cases}$$

The following definitions make precise the terms used to characterize the class of systems approximated by the predictive device.

Time invariant: An operator $\Im$ is *time-invariant* if $Q^d \Im = \Im Q^d \quad \forall d \in \mathbf{Z}$.

Causality: $\Im$ is *causal* if $\mathbf{u}(l) = \mathbf{v}(l) \forall l \leq k \quad \Rightarrow \quad \Im \mathbf{u}(k) = \Im \mathbf{v}(k)$.

Fading Memory: $\Im : l_N^\infty(\mathbf{Z}) \to l^\infty(\mathbf{Z})$ has *fading memory* on a subset $\mathbf{K}_- \subseteq l_N^\infty(\mathbf{Z}_-)$ if there is a decreasing sequence $w : \mathbf{Z}_+ \to (0,1], \lim_{k\to\infty} w(k) = 0$, such that for each $\mathbf{u}, \mathbf{v} \in \mathbf{K}_-$ and given $\varepsilon > 0$ there is a $\delta > 0$ such that

$$\|\mathbf{u}(k) - \mathbf{v}(k)\|_w < \varepsilon \quad \Rightarrow \quad |\Im \mathbf{u}(0) - \Im \mathbf{v}(0)| < \delta$$

Every sequence $\mathbf{u}$ in $l_N^\infty(\mathbf{Z}_-)$ can be associated with a causal extension sequence $\mathbf{u}_c$ in $l_N^\infty(\mathbf{Z})$ defined as:

$$\mathbf{u}_c(k) \equiv \begin{cases} \mathbf{u}(k) & k \leq 0 \\ \mathbf{u}(0) & k > 0 \end{cases}$$

and each time invariant causal operator $\Im$ can be associated with a functional F on $l_N^\infty(\mathbf{Z}_-)$ defined by

$$F\mathbf{u} = \Im \mathbf{u}_c(0)$$

The operator $\Im$ can be recovered from its associated functional F via

$$\Im \mathbf{u}(k) = FPQ^{-k}\mathbf{u} \tag{36}$$

Then, $\Im$ is continuous if and only if F is, so the above equations establish a one to one correspondence between time invariant causal continuous operators and functionals F on $l_N^\infty(\mathbf{Z}_-)$. In the next the definition of the Laguerre system is given. These can be configured in the general filter structure of *FIG. 4* but also have important theoretical properties.

V.4.2 LAGUERRE SYSTEMS

The set of the Laguerre systems is defined in the complex z-transform plane as:

$$L_s^i = \frac{\sqrt{\eta_i}\, z^{-d_i+1}}{z-a_j}\left[\frac{1-a_j z}{z-a_j}\right]^s, \qquad s = 0,1,\cdots,\infty,\ i = 1,\cdots,N$$

where:

$L_s^i(z)$: is the Z transform of $l_s^i(k)$, the s-th order system for the i-th input.

a_i: is the i-th input generating pole, such that $|a_i| < 1$. This pole is selected as $a_i = 1 - \frac{\Delta T}{T_i}$, where T_i is the dominant time constant for the i-th input variable.

d_i: is the time delay associated with the i-th input variable.

η_i: $= 1 - a_i^2$

The whole set of Laguerre systems can be expressed in a state space form that shows a decoupled input form and therefore can be mapped to the general filter structure in *FIG 4*. Each filter unit (**24**) is configured as a single structured $\{\mathbf{A}_i, \mathbf{B}_i\}$ subfilter. The structure of $\mathbf{A}_i$ is a lower triangular matrix, and $\mathbf{b}_i = [1 \quad 0 \quad \cdots \quad 0]^T$.

The key point here is that the representation is decoupled by input. Balancing can be done to decrease the order of the Laguerre systems, and similarity transforms can be done on the Laguerre filters in order to simplify the configuration to utilize the preferred subfilter units. Similarity transformations do not affect the accuracy of the representation and so proving that the use of Laguerre filters decoupled by input approximate any *time invariant, causal, fading memory* system is equivalent to proving the preferred subfilter structure can approximate any such system. The balancing is a practical mechanism to reduce order without degrading performance.

V.4.3 PROOF OF APPROXIMATION ABILITY OF LAGUERRE SYSTEMS

First some preliminary results are stated:

Stone-Weierstrass Theorem (Boyd,1985).

Suppose $\mathbf{E}$ is a compact metric space and $\mathbf{G}$ a set of continuous functionals on $\mathbf{E}$ that separates points, that is for any distinct $\mathbf{u}, \mathbf{v} \in \mathbf{E}$ there is a $G \in \mathbf{G}$ such that $G\mathbf{u} \neq G\mathbf{v}$. Then for any continuous functional F on $\mathbf{E}$ and given $\varepsilon > 0$, there are functionals, $\{G_1^1, \cdots G_{S_1}^1, \cdots, G_1^N, \cdots G_{S_N}^N\} \subseteq \mathbf{G}$, $S = \sum_{i=1}^{N} S_i$ and a polynomial $p : \mathbf{R}^S \to \mathbf{R}$, such that for all $\mathbf{u} \in \mathbf{E}$

$$\left|F\mathbf{u} - p\left(G_1^1\mathbf{u}, \cdots, G_{S_1}^1\mathbf{u}, \cdots, G_1^N\mathbf{u}, \cdots, G_{S_N}^N\mathbf{u}\right)\right| < \varepsilon$$

The reason for the group indexing, which is not necessary for a general statement of the Stone-Weierstrass theorem, will become apparent in Lemma 2 when each block with a Laguerre operator. In addition, three lemmas are necessary before the theorem can be proved.

Lemma 1: $K_- \equiv \left\{\mathbf{u} \in l_N^\infty(Z_-) \middle| 0 < \|\mathbf{u}\| \leq c_1\right\}$, is compact with the $\|\cdot\|_w$ norm.

Proof: Let $\mathbf{u}^{(p)}$ be any sequence in K_-. We will find a $\mathbf{u}^{(0)} \in K_-$ and a subsequence of $\mathbf{u}^{(p)}$ converging in the $\|\cdot\|_w$ norm to $\mathbf{u}^{(0)}$. It is well know that K_- is not compact in $l_N^\infty(\mathbf{Z}_-)$ with the usual supremum norm $\|\cdot\|_\infty$ (Kolmogorov, 1980). For each l, let be $K_-[-l,0]$ the restriction of K_- to $[-l,0]$. $K_-[-l,0]$ is uniformly bounded by c_1 and is composed of a finite set of values, hence compact in $l_N^\infty[-l,0]$. Since $K_-[-l,0]$ is compact for every l, we can find a subsequence $\mathbf{u}^{(p_m)}$ of $\mathbf{u}^{(p)}$ and a $\mathbf{u}^{(0)} \in K_-[-l,0]$ along which $\mathbf{u}^{(p_m)}$ converges:

$$\sup_{-l \leq k \leq 0}\left|\mathbf{u}^{(p_m)}(k) - \mathbf{u}^{(0)}(k)\right| \to 0 \quad as \quad m \to \infty \qquad (37)$$

Now, let $\varepsilon > 0$. Since $w(k) \to 0$ as $k \to \infty$, we can find $m_0 > 0$ a such that $w(m_0) \le \varepsilon / c_1$. Since $\mathbf{u}^{(p_m)}, \mathbf{u}^{(0)} \in K_-$, we have that

$$\sup_{k \le -m_0} \left| \mathbf{u}^{(p_m)}(k) - \mathbf{u}^{(0)}(k) \right| w(-k) \le 2c_1 w(m_0) < \varepsilon \tag{38}$$

Now from equation (37) we can find m_1 such that

$$\sup_{-m_0 < k \le 0} \left| \mathbf{u}^{(p_m)}(k) - \mathbf{u}^{(0)}(k) \right| < \varepsilon \quad for \quad m > m_1 \tag{39}$$

so by equation (38) and equation (39) we can conclude that

$$\left\| \mathbf{u}^{(p_m)} - \mathbf{u}^{(0)} \right\|_w < \varepsilon \quad for \quad m > m_1$$

which proves that K_- is compact.

Lemma 2. The set of functional $\{G_s^i\}$ associated to the discrete Laguerre Operators are continuous with respect to $\|\cdot\|_w$ norm, that is, given any $\varepsilon > 0$ there exists $\delta > 0$ such that

$$\|\mathbf{u} - \mathbf{v}\|_w < \delta \quad \Rightarrow \quad \left| G_s^i \mathbf{u} - G_s^i \mathbf{v} \right| < \varepsilon$$

Proof. Consider the functional $G_s^i(\cdot)$ associated with the Laguerre operator $L_s^i(\cdot)$. Given $\varepsilon > 0$, chose $\delta > 0$ such that:

$$|u_i - v_i|_w < \delta \quad \Rightarrow \quad \left| G_s^i u_i - G_s^i v_i \right| < \varepsilon \tag{40}$$

This is possible due to the continuity of the one dimensional Laguerre operators with respect to the weighted norm as shown in (Sentoni et al, 1996). Therefore, from equation *(40)* and the definition of the functionals

$$\|\mathbf{u} - \mathbf{v}\|_w < \delta \Rightarrow |u_i - v_i|_w < \delta \Rightarrow \left| G_s^i \mathbf{u} - G_s^i \mathbf{v} \right| = \left| G_s^i u_i - G_s^i v_i \right| < \varepsilon \tag{41}$$

which proves Lemma 2

Lemma 3. The $\{G_s^i\}$ separate points in $l_N^\infty(\mathbf{Z}_-)$, that is, for any distinct $\mathbf{u}, \mathbf{v} \in l_N^\infty(\mathbf{Z}_-)$ there is a $G_s^i \in \mathbf{G}$ such that $G_s^i \mathbf{u} \ne G_s^i \mathbf{v}$.

Proof. Suppose $\mathbf{u}, \mathbf{v} \in l_N^\infty(\mathbf{Z}_-)$ are equal except for the i-th component. Then

$$G_s^i\mathbf{u} \neq G_s^i\mathbf{v} \Leftrightarrow G_s^i u_i \neq G_s^i v_i \tag{42}$$

by the definition of the functionals. It is known from one dimensional theory (Sentoni et al, 1996) that for any distinct $u_i, v_i \in l^\infty(\mathbf{Z}_-)$ there is a G_s^i such that $G_s^i u_i \neq G_s^i v_i$; this result together with equation (42) proves Lemma 3.

<u>**Approximation Theorem**</u>

Now given $\varepsilon > 0$, Lemmas 1, 2, 3 together with the Stone-Weierstrass theorem imply that given any continuous functional F on K_-, there is a polynomial $p : \mathbf{R}^S \to \mathbf{R}$, such that for all $\mathbf{u} \in K_-$

$$\left|F\mathbf{u} - p\left(G_1^1\mathbf{u}, \cdots, G_{S_1}^1\mathbf{u}, \cdots, G_1^N\mathbf{u}, \cdots, G_{S_N}^N\mathbf{u}\right)\right| < \varepsilon \tag{43}$$

Because the Laguerre systems are continuous and acting on a bounded space, the $G_s^i\mathbf{u}$ are bounded real intervals on so the polynomial p can be replaced by any static model that acts as a universal approximator on a bounded input space, for example, a neural net. In other words (43) can be replaced by

$$\left|F\mathbf{u} - NN\left(G_1^1\mathbf{u}, \cdots, G_{S_1}^1\mathbf{u}, \cdots, G_1^N\mathbf{u}, \cdots, G_{S_N}^N\mathbf{u}\right)\right| < \varepsilon \tag{44}$$

A time invariant causal operator $\Im$ can be recovered from its associated functional through equation (36) as

$$\Im\mathbf{u}(k) = FPQ^{-k}\mathbf{u}$$

Now let $\mathbf{u} \in K$ and $k \in \mathbf{Z}$, so $PQ^{-k}\mathbf{u} \in K_-$, hence

$$\left|FPQ^{-k}\mathbf{u} - NN\left(G_1^1 PQ^{-k}\mathbf{u}, \cdots, G_{S_1}^1 PQ^{-k}\mathbf{u}, \cdots, G_1^N PQ^{-k}\mathbf{u}, \cdots, G_{S_N}^N PQ^{-k}\mathbf{u}\right)\right| < \varepsilon$$

Since the last equation is true for all $k \in \mathbf{Z}$, we conclude that for all $\mathbf{u} \in K_-$

$$\left\|\Im\mathbf{u} - \hat{\Im}\mathbf{u}\right\| < \varepsilon$$

In other words, it is possible to approximate any nonlinear discrete time invariant operator having fading memory on K, with a finite set of discrete Laguerre systems followed by a single hidden layer neural net. This completes the proof.

V.5 EQUIVALENTS

Although the foregoing details refer to particular preferred embodiments of the invention, it should be understood that the invention is not limited to these details. Substitutions and alterations, which will occur to those of ordinary skill in the art, can be made to the detailed embodiments without departing from the spirit of the invention. These modifications are intended to be within the scope of the present invention.